63-4-2

D1-82-0253

BOEING SCIENTIFIC RESEARCH LABORATORIES

Optimum Redundancy Under Multiple Constraints

Frank Proschan

T.A. Bray

Mathematics Research

May 1963

D1-82-0253

OPTIMUM REDUNDANCY UNDER MULTIPLE CONSTRAINTS

by

Frank Proschan and T. A. Bray

Mathematical Note No. 298

Mathematics Research Laboratory

BOEING SCIENTIFIC RESEARCH LABORATORIES

May 1963

1. <u>Introduction</u> Kettelle (1962) presents an algorithm for allocating redundancy so as to maximize system reliability without exceeding a specified cost (or equivalently, achieve a specified reliability at minimum cost). In the present paper we develop an algorithm to solve the more general problem of maximizing system reliability without exceeding any of several linear constraints.

Specifically, we consider a system consisting of k "stages". The system functions if and only if each stage functions. Stage i consists of n_i (to be determined) units of type i in parallel, so that stage i functions if and only if at least one of the n_i units of type i function, $i = 1,2,\ldots,k$. Suppose unit i has a "cost" c_{ij} of the j^{th} type, $i = 1,\ldots,k; j = 1,\ldots,r$. As an example, the first type of cost might be weight, the second volume, and the third money. A linear constraint exists on each cost as follows:

$$\sum_{i=1}^{k} c_{ij} n_i \leq c_j, \qquad j = 1,2,\ldots,r. \tag{1}$$

Thus in the example the total weight of the system might be required not to exceed a specified amount c_1, the total volume required not to exceed c_2, and the total cost in dollars required not to exceed c_3.

A unit of type i has probability p_i of functioning, independently of the functioning or non-functioning of the other units of the system. Thus system reliability $P(\underline{n})$, where $\underline{n} = (n_1,\ldots,n_k)$, is given by

$$P(\underline{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i}), \tag{2}$$

where $q_i = 1 - p_i$. Our problem is to choose $\underline{n}$ (a vector of positive integers) so as to maximize $P(\underline{n})$ in (2), subject to constraints (1).

2. Domination Let $c_j(\underline{n}) = \sum_{i=1}^{k} c_{ij}n_i$ represent the j^{th} cost of the redundancy allocation $\underline{n}$. Then $\underline{n}^1$ is said to dominate $\underline{n}^2$ if $c_j(\underline{n}^1) \leq c_j(\underline{n}^2)$, $j = 1,\ldots,r$, while $P(\underline{n}^1) \geq P(\underline{n}^2)$. If, in addition, at least one inequality is strict, then $\underline{n}^1$ is said to dominate $\underline{n}^2$ strictly. A sequence S of redundancy allocations $\underline{n}^h$, $h = 1,2,\ldots$, satisfying the constraints (1), is said to be a dominating sequence if no $\underline{n}^h$ is strictly dominated, and if every $\underline{n}$ satisfying the constraints (1) which is not strictly dominated occurs in S.

It is clear that to solve our problem we need only consider the members of the dominating sequence S. Specifically, we seek that allocation of S with maximum reliability $P(\underline{n})$ among the members of S.

Procedure for Two-Stage System First, to generate the dominating sequence corresponding to a system consisting only of stages 1 and 2, we set up a two-way table in which the entry in row n_1, column n_2 consists of the vector $(c_1(n_1,n_2),c_2(n_1,n_2),\ldots,c_r(n_1,n_2),Q(n_1,n_2))$, where $c_j(n_1,n_2) = c_{1j}n_1 + c_{2j}n_2$, $j = 1,\ldots,r$ and $Q(n_1,n_2) = 1 - (1-q_1^{n_1})(1-q_2^{n_2}))$. This is the vector of costs and unreliability resulting from using n_1 units of type 1 and n_2 units of type 2. Only entries satisfying the constraints (1) are included. We then eliminate from the table any strictly dominated vector, that is any vector all of whose coordinates are at least as large as the corresponding coordinates of some other vector in the table, with strict inequality for at least one coordinate. The remaining undominated allocations constitute a dominating sequence.

See the worked example of Section 5 to help clarify these steps.

Next we shall show that we can construct the dominating sequence corresponding to an s stage system from the dominating sequence corresponding to a subsystem of $s - 1$ stages. Once this is established we will then be able to construct the dominating sequence for the full k stage system recursively; i.e., first for a subsystem consisting of stages 1 and 2, next combining the resulting dominating sequence with stage 3 to yield the dominating sequence for stages 1,2, and 3 combined, etc., until the dominating sequence for the full k stage system is obtained. The following procedure includes the Procedure for Two-Stage System as a special case.

Procedure for s Stage System (called Procedure for short) Set up a two-way table in which the $n_s{}^{th}$ row corresponds to n_s units of type s, while the h^{th} column corresponds to $\underline{n}^h$, the h^{th} member of the dominating sequence for the first $s - 1$ stages. The entry at row n_s, column h, is the vector $(c_1(\underline{n}^h,n_s),c_2(\underline{n}^h,n_s),\ldots,c_r(\underline{n}^h,n_s),Q(\underline{n}^h,n_s))$, the vector of costs and unreliability resulting from using the vector $(\underline{n}^h,n_s)$. Note that $c_j(\underline{n}^h,n_s) = c_j(\underline{n}^h) + c_{sj}n_s$, $j = 1,\ldots,r$, while $Q(\underline{n}^h,n_s) = 1 - (1 - Q(\underline{n}^h))(1 - q_s^{n_s})$. Only entries satisfying the constraints are included. We eliminate from the table any strictly dominated vector (strictly dominated by some other vector in the table). The remaining entries constitute the dominating sequence for stages 1,2,...,s, as we prove in

Theorem 1 The vectors that remain strictly undominated in the

two-way table generated in the Procedure constitute a dominating sequence for the s-stage system.

Proof: We need to prove (a) the allocations obtained following the Procedure include all strictly undominated allocations, (b) every allocation obtained using the Procedure is strictly undominated.

We prove (a) inductively. First note that for a single stage system, all allocations are strictly undominated. Assume then that the allocations obtained by the Procedure for a j stage system, where $j = 1,2,\ldots,s-1$, include all strictly undominated allocations satisfying (1). Consider any allocation $\underline{n} = (n_1,\ldots,n_s)$ satisfying (1). Then by inductive hypothesis $(n_1,\ldots,n_{s-1})$ is dominated by some strictly undominated allocation $(n_1^*,\ldots,n_{s-1}^*)$ obtained by the Procedure. Thus by definition $Q(n_1,\ldots,n_{s-1}) \geq Q(n_1^*,\ldots,n_{s-1}^*), c_j(n_1,\ldots,n_{s-1}) \geq c_j(n_1^*,\ldots,n_{s-1}^*)$, $j = 1,\ldots,r$. It follows that $Q(\underline{n}) = 1 - P(n_1,\ldots,n_{s-1})P(n_s) \geq 1 - P(n_1^*,\ldots,n_{s-1}^*)P(n_s^*) = Q(\underline{n}^*)$, where $n_s^* = n_s$, and that $c_j(\underline{n}) = c_j(n_1,\ldots,n_{s-1}) + c_j(n_s) \geq c_j(n_1^*,\ldots,n_{s-1}^*) + c_j(n_s^*) = c_j(\underline{n}^*)$, $j = 1,\ldots,r$, so that $\underline{n}$ is dominated by $\underline{n}^*$. On the other hand, $\underline{n}^*$ being an entry in the two-way table generated by the Procedure is itself dominated by an allocation obtained following the Procedure. Thus we have proved that every allocation satisfying (1) is dominated by some allocation generated following the Procedure. Hence the inductive proof of (a) is completed.

To prove (b), suppose $\underline{n}^o$ is an allocation obtained using the Procedure. If $\underline{n}^o$ is strictly dominated by any allocation satisfying (1) it must also be strictly dominated by some undominated allocation

satisfying (1). But we have just proved that all undominated allocations satisfying (1) are obtained by the Procedure. Thus $\underline{n}^o$ is strictly dominated by say $\underline{n}^1$ also obtained by the Procedure. This is a contradiction since no allocation obtained under the Procedure can dominate any other allocation obtained under the procedure.

3. Approximations In applications of the Procedure we may generally apply the following approximation. Instead of using

$$Q(n_1,n_2) = 1 - (1 - q_1^{n_1})(1 - q_2^{n_2}) = q_1^{n_1} + q_2^{n_2} - q_1^{n_1}q_2^{n_2},$$

we disregard the product term and use

$$Q(n_1,n_2) \cong q_1^{n_1} + q_2^{n_2}.$$

Similarly, for an s stage system, we approximate (where $\underline{n} = (n_1,\ldots,n_{s-1})$:

$$Q(\underline{n},n_s) \cong Q(\underline{n}) + q_s. \tag{3}$$

Kettelle (1962) shows that the use of this approximation for the case $r = 1$ results in an error in P, the system reliability achieved, of at most Q^2 (where $P + Q = 1$). For the present case of $r > 1$, the proof goes through just as in Kettelle (1962). We do not repeat the details.

In any applications of the Procedure we will use approximation (3) throughout.

Another approximation that may reduce the length of dominating sequences is the following. In comparing a pair of entries in the table developed following the Procedure we may introduce a tolerance factor

ε_j for the j^{th} type of cost and/or a tolerance factor ε_q for unreliability. If two entries in the table differ by ε_j or less in the j^{th} type of cost, they are considered alike as far as the j^{th} type of cost is concerned; similarly if they differ by ε_q or less in unreliability. The result is that domination becomes more likely so that the lengths of dominating sequences are reduced. Problems that might be otherwise unsolvable because of excessively long dominating sequences can sometimes be solved (approximately) by introducing one or more tolerance factors. The most conservative procedure is first to try to solve the original problem without tolerance factors. Then if the dominating sequences are too long to permit a solution, introduce a small tolerance factor ε_q on the unreliability. If the dominating sequences are still too long either increase ε_q or introduce additional tolerance factors ε_j. Continue increasing or adding tolerance factors until a solution is attained.

4. <u>Starting Values for the</u> n_i. As we shall see later the lengths of the dominating sequences determine the limiting size of the problems that can be handled on a computer and the time required to compute solutions. It is therefore of utmost importance that the lengths of the dominating sequences be kept as small as possible. One way to reduce their size is to use as large a starting value for each n_i as is conveniently possible.

We now describe a method for finding such large starting values. (1) Add one unit of each component type in succession until finally a constraint will be violated upon the next addition. (2) Compute the

reliability P of the resulting system. (3) Determine n_i, the minimum number n of units of type i required to achieve a reliability of P or greater, from

$$P \leq 1 - q_i^n. \qquad (4)$$

Then it is clear that the solution to the allocation problem will require a value of n_i at least as large as n_i^o. (4) Thus the starting value of n_i may be taken as n_i^o.

To illustrate the value of this starting procedure, note that in one problem involving 10 stages and 3 constraints using this procedure reduced the length of the dominating sequence from the starting point until a constraint was violated from 334 to 62. In a second problem involving 20 stages and 3 constraints, without this procedure the computation had to be halted at the 10th stage with the length of the dominating sequence 559, while the use of the procedure led to a solution with the final dominating sequence of length 69.

An alternate method for generating starting values for the n_i is to use tolerance factors as described in Section 3 to obtain an approximate solution. After the approximate solution is obtained, use steps (2),(3), and (4) above.

5. <u>Example</u> We illustrate the application of the algorithm with the following worked example, based on one that appeared in Kettelle, 1962. Consider a four-stage system with unit costs and unreliabilities as

follows:

Stage, i	1	2	3	4	Constraint
Cost, c_{i1}	1.2	2.3	3.4	4.5	47.0
Weight, c_{i2}	1	1	1	1	20
Unreliability, q_i	.2	.3	.25	.15	

We wish to choose (n_1,n_2,n_3,n_4) so as to maximize system reliability (2) explicitly given by

$$P(n_1,n_2,n_3,n_4) = (1 - .2^{n_1})(1 - .3^{n_2})(1 - .25^{n_3})(1 - .15^{n_4}), \tag{5}$$

subject to constraints (1) explicitly given by

$$\begin{aligned} &1.2n_1 + 2.3n_2 + 3.4n_3 + 4.5n_4 \le 47.0, \\ &n_1 + n_2 + n_3 + n_4 \le 20. \end{aligned} \tag{6}$$

First we shall obtain starting values for the n_i following the procedure of Section 4. Starting with $n_1 = 1, n_2 = 1, n_3 = 1, n_4 = 1$, we add one component at a time until adding an additional component would violate a constraint. As shown in Table 1 we arrive at a system composition (5,4,4,4) with system reliability .9872. Solving for n_1^o, the minimum n satisfying

$$1 - .2^n \ge .9872,$$

we obtain $n_1^o = 3$. Similarly $n_2^o = 4, n_3^o = 4, n_4^o = 3$, as shown in Table 2.

Table 1 - Computation to Find an Attainable Reliability before Violating a Constraint

Stage 1	Stage 2	Stage 3	Stage 4	Cost, j=1	Weight j=2	
1	1	1	1	11.4	4	
2	1	1	1	12.6	5	
2	2	1	1	14.9	6	Constraint 1 = 47.0
2	2	2	1	18.3	7	
2	2	2	2	22.8	8	Constraint 2 = 20
3	2	2	2	24.0	9	
3	3	2	2	26.3	10	
3	3	3	2	29.7	11	
3	3	3	3	34.2	12	
4	3	3	3	35.4	13	
4	4	3	3	37.7	14	
4	4	4	3	41.1	15	
4	4	4	4	45.6	16	
5	4	4	4	46.8	17	← Attainable Reliability =.9872
5	5	4	4	49.1	18	

Table 2 - Computation of Minimum No. of Each Stage Required to Achieve Attainable Reliability of .9872 for That Stage Alone

Stage 1		Stage 2		Stage 3		Stage 4	
No.	Rel.	No.	Rel.	No.	Rel.	No.	Rel.
1	.8000	1	.7000	1	.7500	1	.8500
2	.9600	2	.9100	2	.9375	2	.9775
3	.9920	3	.9720	3	.9844	3	.9966
		4	.9919	4	.9961		

Next, following the Procedure of Section 2, we obtain the dominating sequence for Stages 1 and 2 combined, as shown in detail in Table 3.

Table 3 - Dominating Sequence for Stages 1 and 2

		Stage 1					
	c_1 c_2 Q	3.6 3 .0080	4.8 4 .0016	6.0 5 .00032	7.2 6 .000064	8.4 7 .000013	
Stage 2	9.2 4 .0081	12.8 7 .0161	14.0 8 .0097	15.2 9 .00842	~~16.4~~ ~~10~~ ~~.008164~~	~~17.6~~ ~~11~~ ~~.008113~~	
	11.5 5 .00243	~~15.1~~ ~~8~~ ~~.01043~~	16.3 9 .00403	17.5 10 .00275	~~18.7~~ ~~11~~ ~~.002494~~	~~19.9~~ ~~12~~ ~~.002443~~	
	13.8 6 .000729	~~17.4~~ ~~9~~ ~~.008729~~	18.6 10 .002329	19.8 11 .001049	21.0 12 .000793	~~22.2~~ ~~13~~ ~~.000742~~	
	16.1 7 .00022	~~19.7~~ ~~10~~ ~~.00822~~	~~20.9~~ ~~11~~ ~~.00182~~	22.1 12 .00054	23.3 13 .000284	24.5 14 .000233	
							etc.

Note that we have listed the costs and unreliability starting with $n_1^o = 3$ and adding one unit at a time for Stage 1 across the top. Starting with $n_2^o = 4$ and adding one unit at a time we list the costs and unreliability for Stage 2 down the side. We obtain entries in the body of the table by adding the respective costs and unreliabilities; only entries satisfying the constraints are retained. Proceeding systematically, comparing pairs of entries, we eliminate all strictly

dominated entries. Thus the entry in row 1, column 4 is eliminated since it is dominated by the entry in row 2, column 2. (Note that each figure in the latter position is less than the corresponding figure in the former position.) Similarly the entries shown with a line through them are strictly dominated. The remaining entries are not strictly dominated. Note that only a portion of the complete table is shown; actually entries continue to be made until a constraint is violated.

In Table 4 the dominating sequence is obtained for Stages 1, 2, and 3 combined. Across the top of Table 4 are listed the entries of the dominating sequence of Stages 1 and 2 obtained in Table 3. Down the side are listed entries corresponding to adding one unit of type 3 at a time starting with $n_3^o = 4$. Only entries satisfying the constraints are listed in the body of the table. Again proceeding systematically comparing pairs of entries, we strike out all dominated entries. Thus the entry in row 2, column 2 is eliminated since it is dominated by the entry in row 1, column 4. The remaining entries constitute an undominated sequence.

Finally we form Table 5 to yield the dominating sequence for Stages 1, 2, 3, and 4 combined. Proceeding as before we obtain the dominating sequence for the full system. The solution to our problem is the entry in the table with lowest unreliability, namely the entry with costs $c_1 = 46.8$, $c_2 = 17$, and unreliability .00840. To obtain the corresponding system composition we must trace back through Tables 5,

Table 4 - Dominating Sequence for Stages 1,2, and 3

		Stages 1, 2										
	c_1 c_2 Q	12.8 7 .0161	14.0 8 .0097	15.2 9 .00842	16.3 9 .00403	17.5 10 .00275	18.6 10 .002329	19.8 11 .001049	21.0 12 .000793	22.1 12 .00054	23.3 13 .000284	24.5 14 .000233
Stage 3	13.6 4 .0039	26.4 11 .0200	27.6 12 .0136	28.8 13 .01232	29.9 13 .00793	31.1 14 .00665	32.2 14 .006229	33.4 15 .004949	34.6 16 .004693	35.7 16 .00444	36.9 17 .004184	38.1 18 .004133
	17.0 5 .00097	29.8 12 .01707	31.0 13 .01067	32.2 14 .00939	33.3 14 .00500	34.5 15 .00372	35.6 15 .003299	36.8 16 .002019	38.0 17 .001763	39.1 17 .000151	40.3 18 .001254	41.5 19 .001203
	20.4 6 .00024	33.2 13 .01634	34.4 14 .00994	35.6 15 .00866	36.7 15 .00427	37.9 16 .00299	39.0 16 .002569	40.2 17 .001289	41.4 18 .001033	42.5 18 .00078	43.7 19 .000524	44.9 20 .000473
	23.8 7 .000061	36.6 14 .016161	37.8 15 .009761	39.0 16 .008481	40.1 16 .004091	41.3 17 .002811	42.4 17 .002390	43.6 18 .001110	44.8 19 .000854	45.9 19 .000601		

Table 5 - Dominating Sequence for Stages 1, 2, 3 and 4

		Stages 1, 2, 3								
	c_1 c_2 Q	26.4 11 .0200	27.6 12 .0136	28.8 13 .01232	29.9 13 .00793	31.1 14 .00665	32.2 14 .00622	33.4 15 .004949	33.3 14 .00500	34.5 15 .00372
Stage 4	13.5 3 .0034	39.9 14 .0234	41.1 15 .0170	42.3 16 .01572	43.4 16 .01133	44.6 17 .01005	45.7 17 .009629	46.9 18 .008349	46.8 17 .00840	~~48.0~~ ~~18~~ ~~.00712~~
	18.0 4 .00051	~~44.4~~ ~~15~~ ~~.02051~~	~~45.6~~ ~~16~~ ~~.01411~~	~~46.8~~ ~~17~~ ~~.01283~~	etc.					
	22.5 5 .000076									

4, and 3. Note that from Table 5 the optimal $n_4 = 3$, while from Table 4, the optimal $n_3 = 5$. From Table 3, the optimal $n_1 = 4$, $n_2 = 5$. Actually in the machine program (see Section 6) the composition of the system is retained at each stage so that no retracing is necessary. Thus our solution consists of a composition of $n_1 = 4, n_2 = 5, n_3 = 5$, $n_4 = 3$, with associated reliability (exact value) from (5) of

$$Q(4,5,5,3) = (1 - .2^4)(1 - .3^5)(1 - .25^5)(1 - .15^3) = .9916.$$

Note that the error using approximation (3) throughout is $\leq .0084^2 = .000071$

6. Computer Program The procedure described in Section 2 above was programmed for the IBM 7090 Data Processing System. The program was written with the capacity to handle one, two, or three cost constraints, a maximum of sixty-four stages, a maximum of ten units for each stage, and a maximum of 1024 entries in the dominating sequence at any combining

step. The essential features of the program are presented in the flow chart of Table 6. The following notation is used in the flow chart:

Input Quantities:

k - no. of stages

r - no. of cost constraints

c_j - value of the j^{th} cost constraint

c_{ij} - j^{th} cost of one unit of the i^{th} stage

p_i - reliability of one unit of the i^{th} stage

ε_q - tolerance for unreliability

ε_j - tolerance for j^{th} cost

Computed Quantities:

q_i - unreliability of one unit of the i^{th} stage

s_i - no. of units of i^{th} stage to start algorithm

P_0 - trial system reliability

H_i - no. of row headings for the i^{th} stage

NC_{ih} - no. of units of the i^{th} stage at the h^{th} column heading

M - no. of column headings

I - index for the stage currently being combined into the system

QCH_h - unreliability of the system represented at the h^{th} column heading

QRH_h - unreliability of the no. of units of the I^{th} stage represented at the h^{th} row heading

CCH_{jh} - j^{th} cost of the system represented at the h^{th} column heading

CRH_{jh} - j^{th} cost of the no. of units of the I^{th} stage represented at the h^{th} row heading

TE_{ij} - indicator for the table entry at the i^{th} row and j^{th} column (0 indicates entry is in the dominating sequence; 1 indicates entry is not in the dominating sequence)

TNC_{ih} - temporary storage for the no. of units of the i^{th} stage at the h^{th} column heading

The circled numbers appearing by the boxes of the flow chart refer to the following explanatory notes:

1. The input quantities required are entered into the computer through the appropriate input device.
2. The unreliability of one unit of each stage is computed. The s_i are initialized to one.
3. This loop adds one unit of each stage successively until one of the cost constraints is exceeded.
4. The last s_i to which a unit was added is reduced by one so that the system represented by s_i units of the i^{th} stage, $i = 1,\ldots,k$, is one which violates no constraints.
5. The trial reliability, P_0, of this system is computed.
6. The s_i are initialized to zero.
7. This loop computes the number of units, s_i, of each stage required so as to make the reliability of that stage at least equal to the trial reliability, P_0. The resulting s_i are the values used to start the algorithm.
8. The number of row headings for the i^{th} stage is computed to be the largest integer, H_i, such that $H_i \leq$ the maximum number of units permitted for each stage, and $\sum_{n=1}^{H_i} n \cdot c_{ij} \leq c_j$ for $j = 1,\ldots,r$.
9. The units count for the first stage, NC_{1h}, is set at each column heading, h.
10. The index, I, of the stage to be added to the system is set to two. The number of column headings, M, is set.
11. The unreliability of the system represented at each column heading is computed.

12. The costs of the system represented at each column heading are computed.

13. The unreliability of the units represented at each row heading is computed.

14. The costs of the units represented at each row heading are computed.

15. This loop computes the costs that occur at each entry in the table by adding the costs at the respective row and column headings. If any of the costs exceed a cost constraint the entry is marked as not being in the dominating sequence, otherwise the entry is marked as being in the dominating sequence.

16. This loop steps the indices from table entry to table entry and checks the table entry indicator to see if the entry is in the dominating sequence. If it is not, then the indices are stepped to the next table entry. If it is, then the next loop is executed.

17. This loop compares the table entry just chosen with every other entry in the dominating sequences and marks every entry which it dominates appropriately. It then goes back to continue the loop described in 16 above.

18. When the loop described in 16 above has been exhausted, the table entries marked with a zero are the true dominating sequence. The units counts for the column headings are now moved to temporary storage locations so that the column headings may be updated.

19. This loop updates the units counts for the column headings.

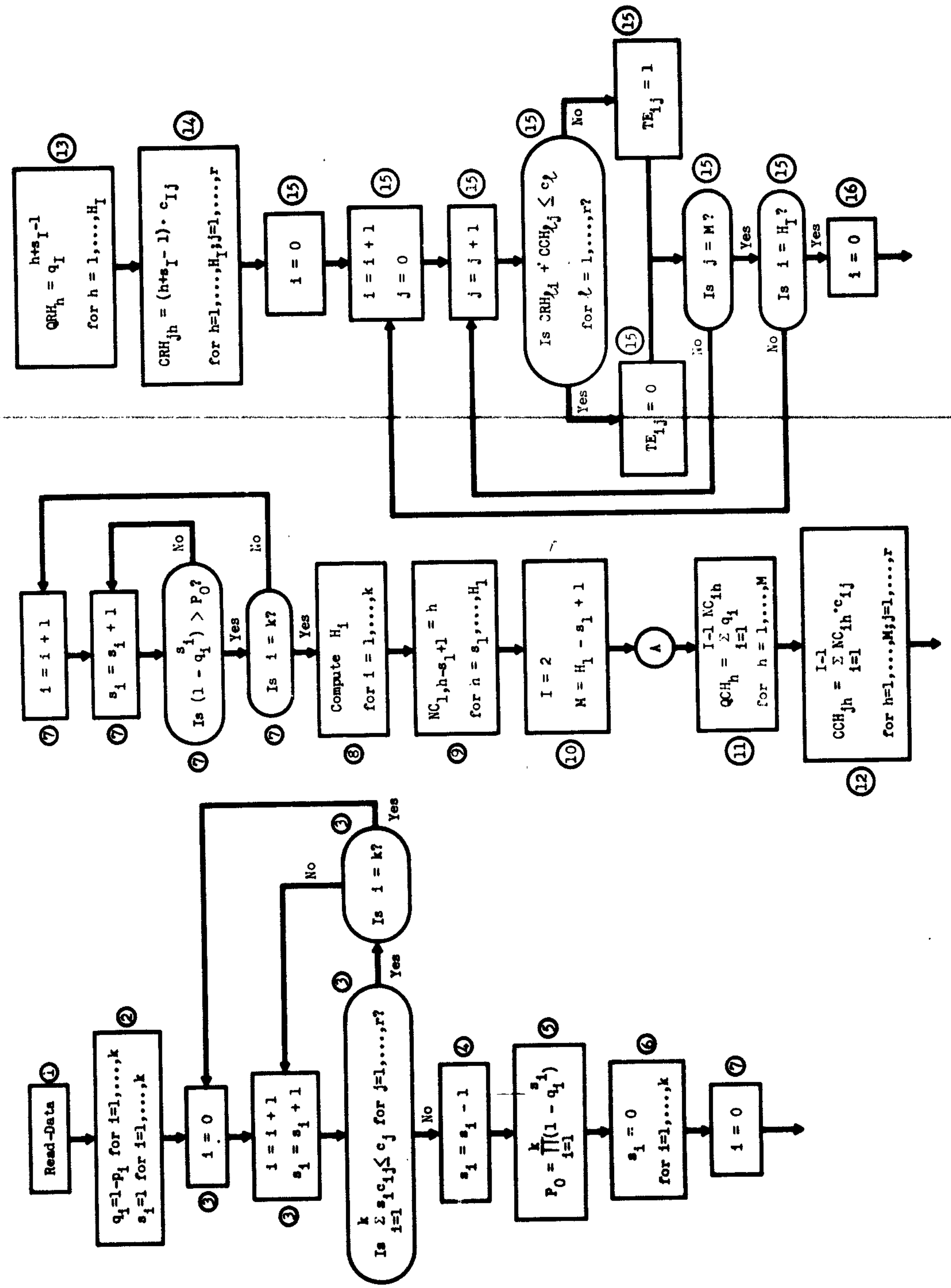
20. If all stages have not been combined into the system, then the index I is stepped for the next stage, the number of column headings is set, and the program is repeated from point A .

21. When all stages have been combined into the system, this loop selects the set of units counts having the maximum reliability. These units counts are the elements of the vector $\underline{n}$ and represent the optimum redundant system.

22. The vector $\underline{n}$ is printed through the appropriate output device.

7. Problem Solving Experience

The purpose of this section is to present some of the experiences which have resulted from attempts to solve problems of varying size and complexity using the computer program described in the previous section. This discussion is included because it is often difficult to predict whether the solution to a given problem is practical without prior experience on a similar problem. Even though the solution to a problem may appear to be practical in the sense that the basic quantities do not exceed the program limitations, it may turn out to be impractical because the dominating sequences developed exceed computer capacity. It is also possible that the computer time required may be excessive. The experience recorded here may provide some guideposts in estimating the practicality of attempting to solve future problems.


Read-Data
q_i = 1-p_i for i=1,...,k
s_i = 1 for i=1,...,k
i = 0
i = i + 1
s_i = s_i + 1
Is Σ s_i c_ij ≤ c_j for j=1,...,r?
Yes
No
Is i = k?
s_i = s_i - 1
P_0 = ∏(1 - q_i^{s_i})
s_i = 0 for i=1,...,k
i = 0
i = i + 1
s_i = s_i + 1
Is (1 - q_i^{s_i}) > P_0?
Is i = k?
Compute H_i for i = 1,...,k
NC_{i,h-s_i+1} = h for h = s_i,...,H_i
I = 2
M = H_1 - s_1 + 1
A
QCH_h = Σ q_i^{NC_ih} for h = 1,...,M
CCH_jh = Σ NC_ih · c_ij for h=1,...,M; j=1,...,r
QRH_h = q_I^{h+s_I-1} for h = 1,...,H_I
CRH_jh = (h+s_I - 1) · c_Ij for h=1,...,H_I; j=1,...,r
i = 0
i = i + 1
j = 0
j = j + 1
Is CRH_ℓi + CCH_ℓj ≤ c_ℓ for ℓ = 1,...,r?
TE_ij = 1
TE_ij = 0
Is j = M?
Is i = H_I?
i = 0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

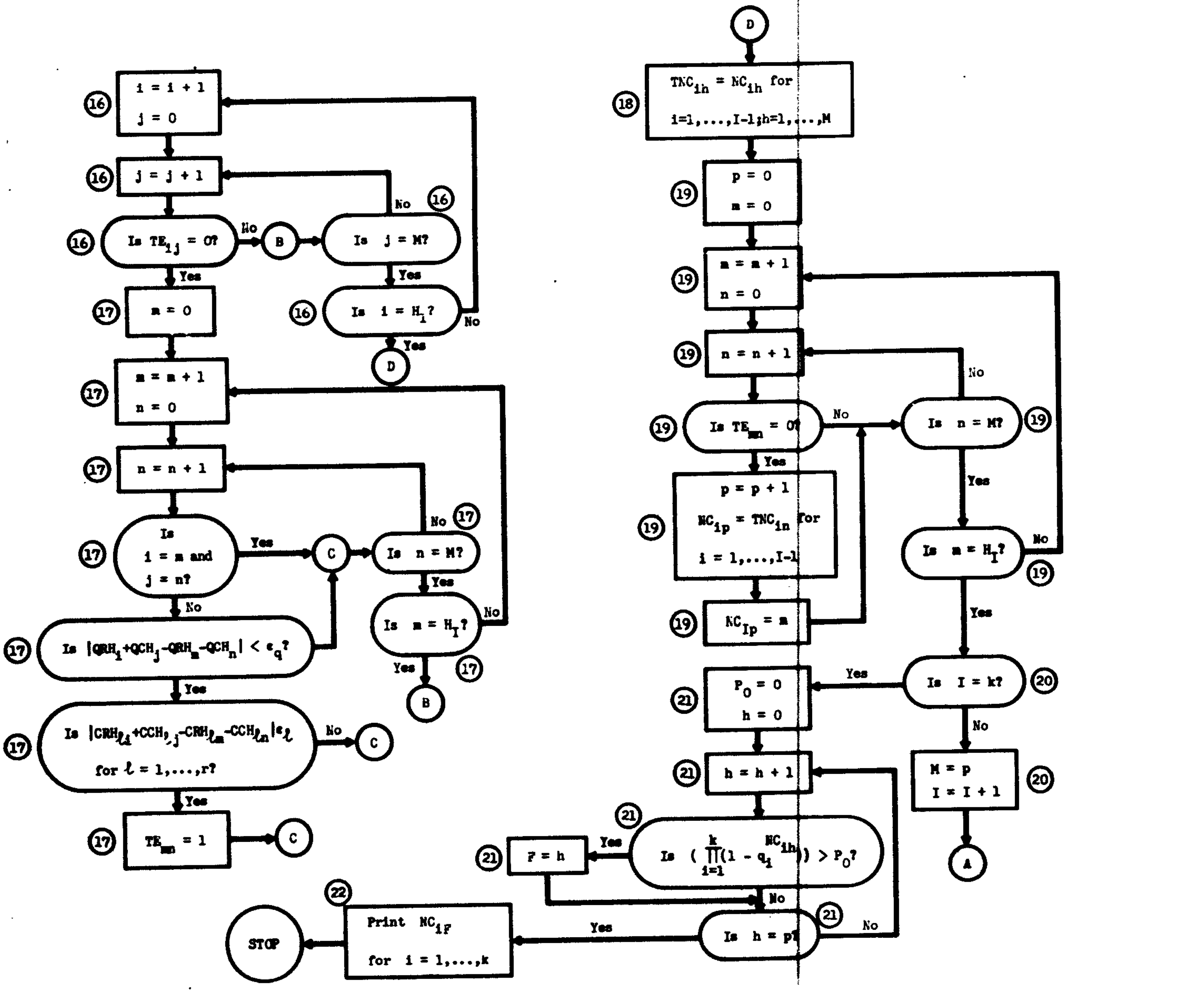


Table 6 - Flow Chart of Computer Program (continued)

A series of problems was devised to test the program systematically. Table 7 presents the basic data for the problems and indicates the nature of the results. One additional problem involving twenty-five stages was attempted but was not successfully completed because the dominating sequences became so large that the computer time required was excessive. This does not necessarily mean that all problems involving more than twenty stages are impractical. The only positive method for determining whether it is practical to solve a given problem is to attempt to find the solution.

Table 7 - Problems and Results

No. of Stages	No. of Constraints	Upper Limit Set on No. of Units of Each Stage	Tolerance Factor, ε	Average Length of Dom. Seq.	Maximum Length of Dom. Seq.	Error in P
10	3	6	0	62	113	0
10	3	6	10^{-7}	35	62	0
10	3	6	10^{-6}	21	47	0
10	3	6	10^{-5}	9	19	0
10	3	6	10^{-4}	2	3	17×10^{-5}
20	2	5	0	198	341	0
20	2	5	10^{-7}	174	155	0
20	2	5	10^{-6}	27	48	63×10^{-8}
20	2	5	10^{-5}	15	21	10^{-5}
20	3	5	0	124	214	0
20	3	5	10^{-7}	98	150	0
20	3	5	10^{-6}	64	112	0
20	3	5	10^{-5}	31	58	78×10^{-7}

8. Other Problems Solvable by the Same Method The method developed above applies to a number of problems other than the redundancy allocation problem.

(1) Spare Parts Kit (See Proschan, 1960.) A system is required to operate during $[0,t]$. When a component fails, it is instantly replaced by a spare, if one is available. The components considered operate independently and are essential to continued system operation, so that a shortage of any component results in system shutdown. Only the spares originally provided may be used for replacement.

Let $P_i(n)$ be the probability that n or fewer spares of type i are required (i.e., n or fewer failures of type i occur during $[0,t]$), $i = 1,\ldots,k$. Then the probability $P(\underline{n})$ of system survival during $[0,t]$ if a spares kit of composition $\underline{n} = (n_1,\ldots,n_k)$ is provided is given by

$$P(\underline{n}) = \prod_{i=1}^{k} P_i(n_i).$$

The problem is to choose $\underline{n}$, a vector of positive integers, so as to maximize $P(\underline{n})$ subject to linear constraints (1).

Note that in the typical application the failure distribution for component type i is often taken to be exponential, $1 - e^{-\lambda_i t}$. In such cases $P_i(n)$ is the Poisson distribution

$$P_i(n) = e^{-\lambda_i t}\left(1 + \lambda_i t + \cdots + \frac{(\lambda_i t)^n}{n!}\right).$$

(2) Optimum Spares Kit when Repair Is Allowed As above, the system is required to operate during $[0,t]$. When a component fails

it is replaced by a spare, if available. Repair is begun immediately on the failed component. There are w_i units of the i^{th} type simultaneously in operation in the system with n_i spares available for replacement, $i = 1,2,\ldots,k$. The failure distribution of a component of type i is exponential, $1 - e^{-\lambda_i t}$, while the repair distribution is arbitrary with mean μ_i, with $w_i\lambda_i\mu_i < 1$. All failure and repair times are independently distributed.

The system is considered to have failed if for any component type no spare is available to replace a failure; i.e., if say for type i, a failure of one of the w_i operating components occurs while all n_i spares are simultaneously under repair. It can be shown (Karush, 1957) that under these assumptions, the steady state probability $P(\underline{n})$ that the system will be "available" (i.e., not shut down due to shortage) is given by

$$P(\underline{n}) = \prod_{i=1}^{k} P_i(n_i),$$

where

$$P_i(n_i) = \sum_{h=0}^{n_i} \frac{(w_i\lambda_i\mu_i)^h}{h!} \Bigg/ \sum_{h=0}^{n_i+1} \frac{(w_i\lambda_i\mu_i)^h}{h!} . \tag{8}$$

As before, the problem is to choose $\underline{n}$ a vector of positive integers so as to maximize $P(\underline{n})$ subject to linear constraints (1). Karush, 1957, shows how to solve (approximately) the problem when a single constraint is present $(r = 1)$.

The algorithm presented above applies to the solution of both problems (1) and (2). The only change is to use the $P_i(n_i)$ appropriate for the particular problem, in carrying out the Procedure of Section 2.

REFERENCES

John D. Kettelle, Jr., "Least-cost allocations of reliability investment", Operations Research, Vol. 10, No. 2, March-April 1962, pp. 249-265.

Frank Proschan, "Optimal system supply", Naval Research Logistics Quarterly, Vol. 7, No. 4, December 1960, pp. 609-646.

William Karush, "A queueing model for an inventory problem", Operations Research, Vol. 5, No. 5, October 1957, pp. 693-703.